Localização de Palavras

Renato Cordeiro Ferreira

3 de junho de 2013

1 Introdução

Neste exercício-programa, tínhamos como objetivo criar uma estrutura que manipulasse as saídas geradas pelo programa CoreNLP, disponibilizado pela Universidade de Stanford[1]. O CoreNLP é um software de processamento de linguagem natural que oferece o recurso de "lematização": para cada palavra presente num texto, geramos o correspondente "lema" - a origem da palavra.

Para este programa, foram utilizadas as estruturas de listas ligadas e árvores rubro-negras esquerdistas para implementar tabelas de símbolos que manejassem palavras e lemas.

2 Arquivos

O presente EP é composto dos seguintes arquivos:

- getline.c
- getline.h
- Item.h
- lemma.c
- lemma.h
- list.c
- list.h

[1] Disponível em http://nlp.stanford.edu/software/corenlp.shtml

- main.c
- Makefile
- ST.c
- ST.h
- word.c
- word.h

Cada um deles apresenta algumas particularidades:

2.1 getline

O arquivo 'getline', retirado das notas de aula do prof. Yoshiharu, foi utilizado como um módulo para leitura de strings de tamanho desconhecido. Com ele, foi possível produzir um buffer que armazenasse o texto lido.

2.2 lemma

Um módulo que utiliza as estruturas de lista e tabela de símbolos para poder construir as funções necessárias para o acesso, contagem e impressão de dados relacionados aos lemas.

2.3 list

Uma lista generalizada, implementada com o auxílio de ponteiros do tipo **void *** e com tipos de 1ª classe. É possível usá-la como uma lista genérica para itens de qualquer tipo.

2.4 main

Arquivo principal do texto que contêm a interface com o usuário (via linha de comando e opções de execução) e a leitura (em fase de pré-processamento) do texto passado como parâmetro ao programa.

2.5 makefile

Arquivo para a compilação dos arquivos. Sua estrutura é de propósito geral e permite compilar projetos diversificados com diretórios.

2.6 ST

Tabela de símbolos implementada com o uso de **ARNEs** (Árvores Rubro-Negro Esquerdistas). Estas árvores foram generalizadas para a utilização como tipo de 1ª classe e para aceitarem qualquer tipo de estrutura (por meio de ponteiros **void ***).

As árvores rubro-negras foram baseadas no código disponibilizado na página do professor Yoshiharu (llrb.c) e modificados para aceitar diferentes tipos de chaves. A função de definição da chave precisa ser especificada juntamente com as funções "less", "eq" e o tipo "NULLitem" na função "STinit()".

2.7 word

Módulo relacionado a funções de manipulação da tabela de símbolos que armazena palavras. Contém estruturas que guardam uma lista de sentenças, o lema e a própria palavra (por meio de ponteiros para um buffer) que auxiliam a imprimir, contar e acessar os dados presentes na tabela de símbolos.

3 Considerações finais

Dada a generalização das estruturas propostas por esta implementação do exercício-programa, o custo do uso de ponteiros **void *** durante quase todo o processo diminuiu a eficiência do pré-processamento e carregamento das estruturas de dados.

Isso não impediu, porém, que o acesso e impressão dos dados fossem realizados de forma eficiênte, dado que os algoritmos de complexidade logarítmica (O(n)) inerentes às ARNEs mantiveram-se assimptoticamente ótimos para a estrutura.